ADVERSARIAL ATTACKS

a Modelli di Machine Learning

Andrea Pasquinucci (PhD - CISA - CISSP)

WHITEPAPER 10/2023

ICT Security MAGAZINE

INDICE

ICT Security MAGAZINE

FORUM ICT SECURITY

25-26 OTTOBRE 2023

AUDITORIUM DELLA TECNICA, ROMA

Iscriviti alla newsletter di ICT Security Magazine
per conoscere l'agenda e partecipare alla
21ª Edizione del Forum ICT Security

ICT SECURITY MAGAZINE

1° rivista italiana di sicurezza informatica, attiva da oltre 20 anni, dedicata in forma esclusiva alla cyber security e alla business continuity, si pone l'obiettivo di coinvolgere i più importanti attori del settore, aziende e istituzioni pubbliche, per la diffusione degli elementi conoscitivi legati a tutti gli aspetti della information security.

ABOUT THE AUTHOR

Andrea Pasquinucci

(PhD - CISA - CISSP)

Consulente freelance in sicurezza informatica: si occupa prevalentemente di consulenza al top management in Cyber Security e di progetti, governance, risk management, compliance, audit e formazione in sicurezza IT.

"*Adversarial Machine Learning*", "*Adversarial Attacks*", "*Adversarial Examples*" e "*Adversarial Robustness*" sono termini che appaiono sempre più spesso quando si considerano aspetti di sicurezza dei modelli di *Machine Learning* (ML). In questo white paper vengono presentati i principali aspetti di quest'ambito, cercando di fare una rassegna di un campo in rapida evoluzione.

Bug Hunting e Adversarial Examples

Come pratica ormai consolidata nella sicurezza informatica, una nuova applicazione, sistema o servizio va prima o poi soggetto a delle verifiche di "sicurezza", intesa anche in senso lato, da parte di esperti che cercano di trovare errori, comportamenti anomali o vere e proprie vulnerabilità. Questo accade anche per i modelli di Intelligenza Artificiale e in particolare di *Machine Learning*.

Sin dalle prime applicazioni pratiche dei modelli ML, nei primi anni 2000, i ricercatori si impegnarono a verificare il loro comportamento: ad esempio trovando modalità di evadere i filtri anti-SPAM basati su modelli ML e aggirare le applicazioni di ML per il riconoscimento di immagini (inizialmente questi erano chiamati genericamente *Evasion Attacks,* mentre il termine maggiormente usato oggi è *Adversarial Examples*).

Queste attività possono essere considerate l'equivalente di eseguire un *Penetration Test* su un'applicazione Web, o di trovare il modo di aggirare o ingannare un sensore biometrico con funzioni di controllo degli accessi. Riportiamo qui solo due esempi provenienti dalla ormai enorme letteratura a riguardo [Rif. 1] e relativi ad applicazioni ML per il riconoscimento di immagini.

Fig. 1 Cartello stradale di Stop con sticker [Fonte Rif. 2]

La Fig. 1 presenta un ben riconoscibile cartello stradale di Stop ma, a causa dello *sticker* giallo apposto sotto la scritta, il modello ML sotto attacco interpreta il segnale con il 94,7% di probabilità come un limite di velocità (tipicamente 40Kmh o 25Mph). Una ipotetica vettura a guida autonoma che utilizzasse questo modello ML per riconoscere la segnaletica stradale, quindi, non si fermerebbe allo Stop ma proseguirebbe a velocità inferiore al limite, con ben immaginabili possibili conseguenze.

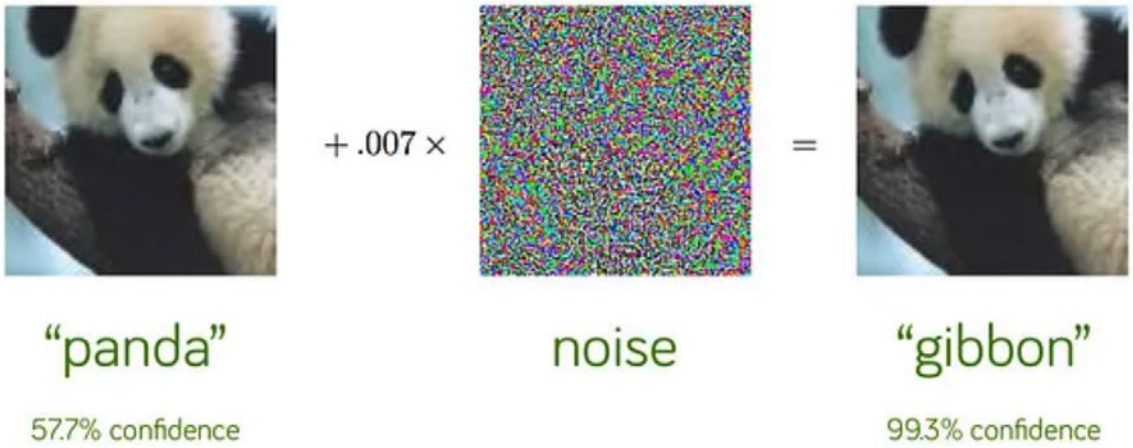


Fig. 2 Un panda o una scimmia gibbone? [Fonte Rif. 3]

La Fig. 2 presenta un'immagine di un panda (a sinistra) in cui al 7 per mille dei pixel viene aggiunto del "rumore bianco". Il risultato è che, per l'occhio umano, l'immagine non è cambiata; mentre per il modello ML ora rappresenta una scimmia gibbone con il 99,3% di probabilità.

Lo studio di queste vulnerabilità ha portato a comprendere – si veda ad esempio [Rif. 4] – che non si tratta di errori dovuti ad una particolare implementazione errata o disattenzione dei programmatori, ma di un problema più generale e profondo dei modelli ML.

Per semplicità e chiarezza di esposizione, conviene distinguere due significati simili ma non precisamente identici del termine "*Adversarial Machine Learning*":

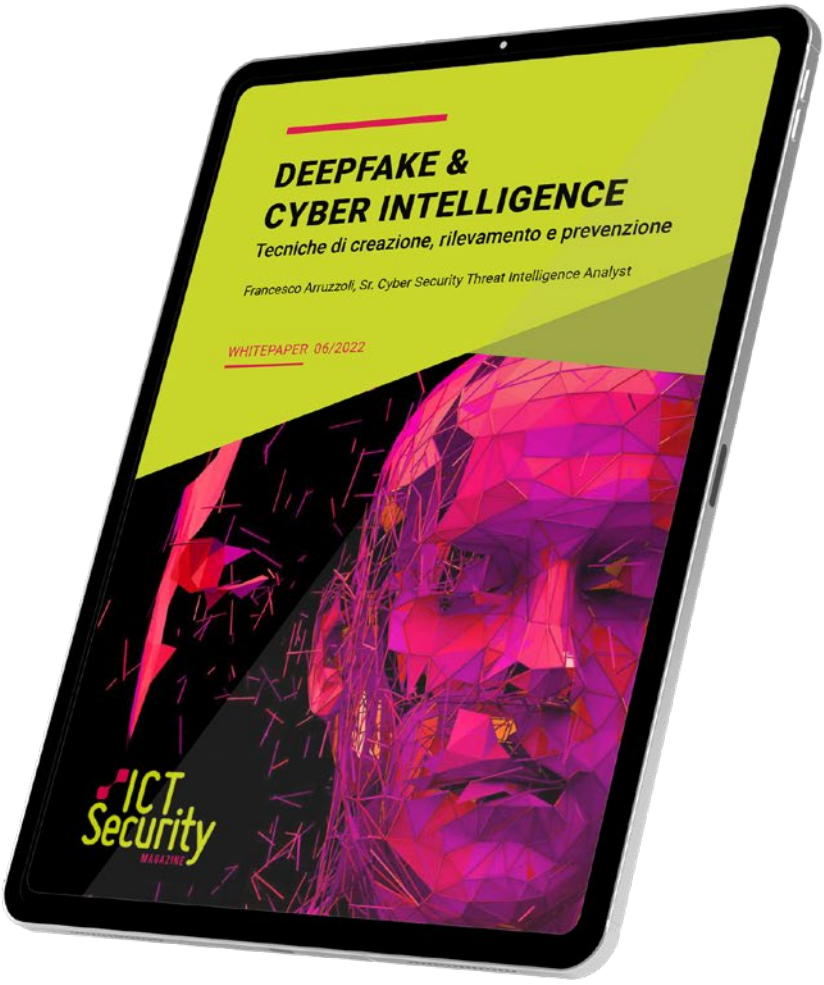
1. ***Adversarial ML* – Aspetti Scientifici**: studio della "Robustezza" dei modelli tramite la possibilità di costruire *Adversarial Examples* con tecniche sperimentali e matematiche – si vedano i due esempi precedenti;
2. ***Adversarial ML* – Aspetti Operativi**: Studio dei possibili attacchi ai modelli ML utilizzati in ambiente Business.

I due ambiti hanno ovviamente ampie aree di sovrapposizione ma i principali obiettivi, pur essendo allineati, sono diversi.

White Paper

DEEPFAKE & CYBER INTELLIGENCE

Download gratuito su www.ictsecuritymagazine.com



Adversarial Machine Learning – Aspetti Scientifici

In ambito scientifico lo scopo della ricerca è comprendere quali sono le ragioni dell'esistenza degli *Adversarial Examples* e, di conseguenza, come migliorare i modelli ML in modo da non averne. I modelli ML senza *Adversarial Examples* sono chiamati **Robusti:** questo non vuol dire che non possono fare errori o confondersi, ma che non hanno deviazioni sistematiche con grandi errori dal comportamento atteso.

Nel primo esempio precedentemente riportato, in un modello ML Robusto l'applicazione di *sticker* o altri camuffamenti e scritte che all'occhio umano non nascondono il cartello stradale di Stop, possono al più ridurre la confidenza dell'identificazione del cartello da parte del modello ML ad una percentuale del 70% o 60%, ma non identificare il cartello come un altro segnale con quasi assoluta certezza. Nel secondo esempio, due immagini che appaiono uguali all'occhio umano devono essere classificate similmente da un modello ML Robusto, al più con diverso livello di confidenza; e non possono essere identificate come raffiguranti due animali diversi, quello errato con quasi assoluta certezza.

È possibile quindi paragonare il processo di *creazione di un modello ML Robusto* al processo di *sviluppo sicuro del software*: ormai è ben noto come sviluppare software adottando pratiche che permettono di ridurre il più possibile la probabilità di *Bugs* e vulnerabilità di sicurezza, anche se rimane molto difficile se non impossibile (anche teoricamente) scrivere del software complesso con garanzia assoluta di assenza di vulnerabilità.

Ma i modelli ML che sono oggi disponibili sono ancora molto giovani. Benché i primi sviluppi di modelli di Intelligenza Artificiale risalgano agli anni '50, lo sviluppo dei modelli ML odierni è iniziato decisamente negli anni '90. Anche se l'architettura di base di molti modelli ML è a prima vista semplice, gli algoritmi matematici che li supportano possono essere molto complessi e ancora non ben compresi/risolti dai ricercatori, ad esempio quelli dei modelli di *Deep Learning* che recentemente hanno fatto tanto clamore giornalistico.

Siamo quindi nel periodo in cui si è identificato un problema (l'esistenza di *Adversarial Examples* in modelli ML) e lo si sta studiando; ma non si è ancora trovata la soluzione generale (come costruire

modelli ML Robusti) né l'approccio operativo per risolverlo.

L'esistenza di *Adversarial Examples* è legata al processo di apprendimento di un modello ML. Una tra le principali caratteristiche dei modelli ML è avere moltissimi, anche centinaia di miliardi di parametri numerici configurabili. Inizialmente questi parametri hanno valori casuali e il processo di apprendimento consiste appunto nel configurare il miglior valore di ogni parametro in modo che ogni dato di addestramento in ingresso venga elaborato dal modello, producendo in uscita il risultato atteso. Semplificando molto il funzionamento di questi modelli, quando un nuovo dato viene elaborato dal modello ML, i valori dei parametri configurati permettono di identificare somiglianze con i dati di addestramento e fornire un risultato probabilisticamente vicino a quello corretto od ottimale. Ovviamente il set di dati di addestramento non è esaustivo di tutti i possibili dati esistenti: nei due esempi precedenti non comprende tutte le possibili immagini di cartelli stradali o di panda, anche perché sarebbe una quantità infinita di dati. I set di dati di addestramento sono però rappresentativi dei dati esistenti ma a priori (e in pratica) non è detto che siano sufficienti a configurare **tutti** i parametri del modello ML in maniera **ottimale**. C'è quindi la possibilità che una particolare combinazione di dati in ingresso corrisponda ad uno speciale set di parametri che non è stato configurato ottimamente per quei dati (o per nulla) e che quindi generi un risultato errato.

Nell'Appendice di questo articolo sono fornite ulteriori indicazioni tecniche su questa problematica.

Una ovvia prima soluzione all'esistenza degli *Adversarial Examples* è addestrare nuovamente il modello ML includendo tra i dati di addestramento gli *Adversarial Examples*. Il primo problema di questo approccio è che bisogna innanzitutto essere capaci di identificare **tutti** gli *Adversarial Examples*; e questo è proprio uno dei principali problemi ancora aperti. Inoltre alcuni modelli ML hanno mostrato di essere soggetti a vulnerabilità o debolezze quali la "sotto-specifica" (*Underspecification*, ovvero il fatto che una minima modifica ai dati di addestramento produce grandi modifiche nel modello) e la "smemoratezza" (*Forgetfulness*, ovvero un aggiornamento del modello con nuovi dati porta il modello a dimenticare parte di quello che aveva già appreso). Vi è quindi il rischio che un ulteriore addestramento di un modello ML, aggiungendo anche gli *Adversarial Examples* tra i dati di addestramento, possa introdurre nuovi *Adversarial Examples*, senza quindi risolvere il problema. Queste e simili fragilità attuali del processo di apprendimento dei modelli ML rendono oggi difficile – se non impossibile – identificare ed eliminare completamente gli *Adversarial Examples* dai modelli ML.

Adversarial Machine Learning – Aspetti Operativi

Gli *Adversarial Examples* illustrati nella sezione precedente non sono di per sé degli eventi di attacco ma, più precisamente, delle **vulnerabilità** che possono essere sfruttate da un attaccante.

Fig. 3 Reese Witherspoon o Russel Crowe? [Fonte Rif. 5]

Si consideri l'esempio riportato in Fig. 3, che riporta a sinistra un'immagine dell'attrice Reese Witherspoon, come tale riconosciuta da uno specifico modello ML studiato in [Rif. 5]. Analogamente l'immagine a destra riporta l'attore Russel Crowe, correttamente identificato dal modello ML. Viene poi sottoposta al modello ML l'immagine centrale di Reese Witherspoon a cui sono stati aggiunti dei particolari occhiali creati appositamente (la versione fisica di questi occhiali costa solo pochi Euro). Questa immagine è però riconosciuta dal modello ML come un'immagine di Russel Crowe e non di Reese Witherspoon.

Come per molti altri tipi di vulnerabilità dei sistemi IT, se questo modello ML fosse utilizzato in un sistema di controllo degli accessi, questa vulnerabilità potrebbe essere sfruttata da un attaccante per impersonare un'altra persona e accedere illegalmente a un sistema o servizio.

La presenza di *Adversarial Examples* non è l'unico tipo di vulnerabilità o l'unico modo per attaccare o abusare di un modello ML. Sono state proposte alcune tassonomie di attacchi e minacce ai modelli ML e in generale ai modelli di Intelligenza Artificiale, tra cui quella di Microsoft [Rif. 6] riassunta in Fig. 4 e quella di MITRE [Rif. 7], riassunta in Fig. 5.

Intentional Failures		**Unintended Failures**
Perturbation Attack Query modification	Adversarial in Physical Domain	Reward hacking Training mismatch with reality
Poisoning Attack	Malicious ML provider	Side effects
Model Inversion	Attack ML supply chain	Distribution shift Usage in environment different from test
Membership Inference	Backdoor ML	Natural Adversarial Unexpected real data
Model Stealing	Exploit SW dependencies	Common corruption Unable to manage perturbations
Reprogramming ML Usage not programmed for		Incomplete testing

Fig. 4 *Microsoft Threat taxonomy - Failure modes in machine learning* [Fonte Rif. 6]

Fig. 5 MITRE ATLAS [Fonte Rif. 7]

Senza addentrarsi in dettaglio in queste tassonomie, è utile approfondire i principali tipi di attacchi che possono sfruttare vulnerabilità dei modelli ML, inclusa quella degli *Adversarial Examples*.

Quaderno di Cyber Intelligence #2

CYBER CRIME

White paper gratuito su www.ictsecuritymagazine.com

Attacchi ai dati di addestramento

La qualità e integrità dei dati utilizzati per l'addestramento di un modello ML sono cruciali per ottenere il comportamento atteso. Per l'addestramento è inoltre spesso necessaria una grande mole di dati, spesso forniti da terze parti.

Come semplice esempio, si consideri il caso di un modello ML utilizzato per identificare messaggi di posta elettronica di SPAM. Un attaccante che vuole organizzare una campagna di SPAM è interessato ad evitare che i propri messaggi siano identificati come SPAM dal modello ML. Per far questo può cercare di inserire i propri messaggi nei dati di addestramento classificandoli come non-SPAM. Oppure può cercare di accedere ai dati di addestramento e modificare la classificazione di alcuni di essi in modo che non risultino più classificati come SPAM. Gli esempi di SPAM che forniscono i dati di addestramento del modello ML sono di norma raccolti da molte sorgenti, per lo più pubbliche: può essere pertanto sufficiente che l'attaccante riesca ad attaccare una componente della *Supply Chain* per sovvertire o "*avvelenare*" (*Poisoning*) il modello ML.

La modifica dei dati d'addestramento fatta alla fonte, presso eventuali terze parti che raccolgono i dati o presso l'utente finale che addestra il modello ML, può portare ad esempio alla creazione di *Backdoor* quali comportamenti nascosti attivabili tramite particolari sequenze di dati in input, o di *Bias* (una distorsione o deviazione sistematica rispetto al risultato atteso) che possono permettere ad un attaccante di utilizzare il modello ML per i propri scopi.

Attacchi al codice dei modelli di Machine Learning

Anche piccole modifiche al codice o all'eseguibile di un modello di *Machine Learning* possono portare a

introdurre comportamenti malevoli come quelli appena descritti. Molto spesso un utente di un modello ML non costruisce da zero il codice del modello ma ottiene un modello già costruito e pre-addestrato da un fornitore, per poi completare l'addestramento con i propri dati. Un attaccante può quindi attaccare questa *Supply Chain* con l'intento di modificare il codice del modello ML in modo che questo, anche se ulteriormente addestrato, continui a presentare delle *Backdoor, Bias* o in generale dei comportamenti malevoli. Nel caso più semplice di un modello ML il cui codice sorgente è pubblico, l'attaccante può ottenere e modificare il codice, addestrare il modello ML e far sì, ad esempio tramite un'intrusione nel server che distribuisce il codice, che l'utente utilizzi il modello ML con il codice modificato.

Attacchi al modello di Machine Learning

In questo caso l'attaccante accede come utente ad un modello ML già addestrato per attaccarlo. L'attaccante può non avere alcuna informazione sul modello (attacco *Black Box*), qualche informazione sui dati di addestramento e/o sulla struttura del modello (attacco *Grey Box*), o completa informazione e disponibilità sia dei dati di addestramento sia del modello stesso (attacco *White Box*). In tutti i casi, l'attaccante non è in grado di modificare i dati di addestramento o il codice del modello sotto attacco se non interagendo con il modello ML stesso.

L'attaccante può quindi utilizzare direttamente il modello ML sotto attacco e analizzare i dati prodotti dal modello a seconda dei diversi input inviati al modello. Questo può permettere all'attaccante di identificare delle vulnerabilità del modello ML che possono permettergli di abusare del modello, ovvero ottenere dei risultati non previsti o a proprio favore (*Inference* e *Model Evasion*). Ad esempio l'attaccante può identificare e sfruttare a proprio favore dei *Bias* nel modello ML, oppure costruire degli *Adversarial Examples* come descritto precedentemente, o ancora identificare dei modi di utilizzo del modello ML non previsti e per i quali il modello ML non è stato specificatamente addestrato. Può essere utile presentare alcuni esempi teorici.

Riprendendo l'esempio precedente di un modello ML utilizzato per identificare messaggi di posta elettronica di SPAM, l'attaccante continua a modificare la formulazione del proprio messaggio sino a che il modello cessa di identificarlo come SPAM. Ovviamente in breve tempo i nuovi messaggi di SPAM non correttamente classificati dal modello ML vengono identificati manualmente, il che richiede un ulteriore addestramento del modello ML aggiungendo i nuovi campioni di SPAM. Ciò porta, anche in questo caso, al ben noto ciclo di attività attaccante-difensore tipico di molti sistemi di sicurezza. Come indicato precedentemente, alcuni modelli ML hanno mostrato una vulnerabilità indicata come "smemoratezza" (*Forgetfulness*): ovvero il fatto che ulteriori aggiornamenti di un modello ML con nuovi dati di addestramento possono portare il modello ML a dimenticare informazioni precedentemente acquisite. Se questo succedesse nel presente esempio, una volta ulteriormente addestrato il modello ML non identificherebbe più alcuni messaggi di SPAM che prima identificava correttamente.

Un altro tipico caso di studio è quello di un modello ML a supporto del processo di erogazione di un mutuo, prestito o finanziamento. L'attaccante, utilizzando il modello, scopre che questo ha dei *Bias* e che alcune classi di richiedenti (identificati per gruppo sociale o geografico, etnia ecc.) sono favorite dal modello ML nell'ottenere l'erogazione a discapito di altre. L'attaccante può quindi formulare la domanda in modo da presentarsi al modello ML come parte di un gruppo favorito e quindi avere maggiori possibilità di ottenere il finanziamento, anche se in realtà non gli sarebbe dovuto.

Altri esempi sono i modelli ML a supporto dei sistemi di controllo accessi e videosorveglianza. Come descritto precedentemente, l'identificazione da parte di un attaccante di *Adversarial Examples* per un modello ML utilizzato a questi scopi può permettere all'attaccante di aggirare un sistema di controllo degli accessi o evitare di essere riconosciuto da un sistema di videosorveglianza.

Invece di abusare dei risultati di un modello ML, un attaccante può essere interessato al modello ML stesso: alla sua architettura, al suo codice sorgente e ai dati con cui è stato addestrato. Sempre interagendo con un modello ML, l'attaccante può quindi cercare di estrarre informazioni interne al modello stesso. Come esempio di questo tipo di attacco si può considerare non un modello ML per la classificazione di immagini ma un *ChatBot* come il famoso ChatGPT. Questi tipi di modelli ML interagiscono con l'utente e producono tipicamente testi o immagini. Con opportune richieste al modello ML e analizzando le risposte ottenute, l'attaccante può dedurre e in alcuni casi ottenere copia di alcuni dati utilizzati per l'addestramento del modello, che possono anche essere riservati. Alcuni di questi modelli ML apprendono anche dalle richieste e dalle interazioni con gli utenti; e analogamente

un attaccante potrebbe venire a conoscenza di informazioni fornite in input al modello ML da un altro utente, in alcuni casi violandone la privacy. Sempre con opportune richieste al modello ML, l'attaccante potrebbe venire a conoscenza dell'architettura, configurazione e anche del valore di alcuni parametri. Si tratta quindi di attacchi il cui scopo è l'inversione del modello ML e/o l'estrazione o furto di informazioni dal modello stesso.

Infine alcuni di questi attacchi possono anche essere sfruttati per rendere il modello non operativo, nel senso che il modello non è più in grado di svolgere il compito per cui è stato costruito e addestrato. Ad esempio, nel caso di un modello ML per il riconoscimento di immagini o messaggi di posta elettronica di SPAM, il modello dopo l'attacco non riconosce più alcuna immagine o messaggio di SPAM. Si può quindi classificare l'effetto di questi attacchi come un particolare tipo di *Denial of Service*.

Vulnerabilità e Sicurezza Proattiva

Come già accennato all'inizio di questo articolo, per gestire la "sicurezza" dei modelli di *Machine Learning* è possibile adottare l'approccio usuale alla gestione della sicurezza IT (o Cyber-sicurezza). Anche se di tipo e con caratteristiche diverse, i modelli di ML e in generale di Intelligenza Artificiale sono applicazioni IT e, come tali, possono avere delle vulnerabilità specifiche che possono essere sfruttate da un attaccante. Chi ne gestisce la sicurezza può quindi sottomettere a verifica queste applicazioni prima che siano messe in produzione per individuare l'eventuale presenza di queste vulnerabilità. Verifiche simili – in termini di approccio – a quelle svolte con i *Penetration Test* (o *Red Teaming*) per un'applicazione Web prima del rilascio in produzione, dovrebbero essere fatte anche per i modelli di *Machine Learning* al fine di identificare l'eventuale presenza di vulnerabilità quali quelle brevemente descritte in questo articolo.

Appendice: alcuni aspetti tecnici degli Adversarial Examples

Il processo di addestramento di un modello di *Machine Learning* può essere descritto ad alto livello come composto da un modello ML di cui si è decisa l'architettura, di un set di parametri inizialmente ignoti e di un set di dati di addestramento. I parametri del modello ML sono inizialmente scelti con valori casuali e il processo di addestramento consiste nel calcolare il valore migliore di questi parametri (i modelli di *Deep Learning* hanno un elevato numero di parametri, ad esempio GPT-3/ChatGPT ne ha 175 miliardi). Per far questo si esegue il modello ML sui dati di addestramento, di cui si conosce il risultato atteso, e si misura l'errore del risultato dell'esecuzione del modello ML rispetto al risultato corretto. Inizialmente questo errore è molto grande, visto che il modello ML ha parametri casuali, ovvero non sa nulla. Poi si modificano i parametri in modo da ridurre l'errore.

Per semplicità si consideri un modello ML basato su Reti Neurali, addestrato con dati etichettati (*Supervised Learning*) e si indichi con {W} il set dei parametri del modello e con {X} il set dei dati di addestramento. L'errore del modello ML sui dati di addestramento è quindi una funzione di entrambi, E({W},{X}). Scopo dell'addestramento è cambiare il valore numerico dei parametri in modo da minimizzare l'errore, ovvero trovare $\min_{\{W\}}E(\{W\},\{X\})$.

Per trovare il migliore valore dei parametri, si utilizza principalmente il "metodo del gradiente" (*Gradient Descent*, Cauchy, 1847) o metodi simili. Questo è un metodo iterativo facilmente implementabile numericamente e quindi in software. Il metodo procede facendo piccolissime modifiche {δW} ai parametri, {W}→{W+δW}, tali che per ogni passo viene scelta la modifica che riduce maggiormente l'errore. Cauchy dimostrò che $\delta W=-\varepsilon D_{\{W\}}E(\{W\},\{X\})$, ove ε è un numero piccolo (ad esempio 0,001) e $D_{\{W\}}$ è la derivata della funzione errore rispetto ai parametri del modello ML. Il ciclo di calcolo (Epoca) su tutti i dati di addestramento viene ripetuto sino a quando non si raggiunge un errore molto piccolo, ad esempio inferiore all'1%.

Se è possibile addestrare il modello in modo che produca minimi errori, com'è possibile che vi siano *Adversarial Examples*? Bisogna ricordare che un modello ML apprende utilizzando uno specifico set di dati di addestramento che inizialmente non comprende gli *Adversarial Examples*. Il modello ML è capace

di elaborare correttamente moltissimi dati che sono simili a quelli del set di addestramento, ma non tutti. Esistono particolari dati molto simili a quelli di addestramento sui quali il modello ML compie grandi errori: e proprio questi sono gli *Adversarial Examples*.

In semplici modelli di studio è possibile avere un'idea dell'origine degli *Adversarial Examples* [Rif. 8].

È possibile utilizzare un approccio simile a quello adottato per l'addestramento per cercare e costruire degli *Adversarial Examples*. Si assuma per semplicità di esposizione di aver calcolato i migliori parametri del modello ML e di non modificarli più. Invece si procede a modificare di poco i **dati** di addestramento cercando la modifica che genera il massimo errore, in formula $\max_{\{\delta x\}} E(\{W\},\{X+\delta X\})$, ove δX deve rimanere molto piccolo. Un semplice esempio di cosa si intende per "piccola modifica" può essere fatto per un modello ML che classifica immagini. Per il modello ML ogni immagine è una lunga lista di numeri, ogni numero corrisponde al colore di un pixel dell'immagine (questo se l'immagine è in toni di grigio, per immagini a colori ogni numero è il colore di un canale, ad esempio RGB, di un pixel). Una "piccola modifica" può essere ad esempio modificare di poco il valore di un pixel e solo per 1 numero/pixel ogni 200 (cioè il 5 per mille); in molti casi questo tipo di modifica non è percettibile dall'occhio umano. In altre parole, si modificano alcuni pixel di un'immagine in modo che per l'occhio umano non vi sia alcuna differenza con l'immagine originale, ma in modo che il modello ML interpreti la nuova immagine come qualcosa di completamente diverso, ovvero massimo errore.

Sebbene il punto di partenza sia molto simile al processo di addestramento, la matematica necessaria per risolvere questo problema è molto più complessa e una soluzione completa non è ancora nota.

Riportiamo i risultati di un esperimento di [Rif. 8] che utilizza un semplice modello *Convolutional Neural Network* per il riconoscimento di immagini e per il quale i ricercatori sono stati in grado di risolvere anche solo approssimativamente l'equazione che definisce gli *Adversarial Examples*. Il modello ML è addestrato su di un set di dati di addestramento (*Train*) e verificato su di un set di dati di prova (*Test*) su cui non è addestrato. Per ogni ciclo di addestramento (Epoca) viene anche calcolato un set di *Adversarial Examples* (*Adv*), ovvero il set $\{X_{adv}=X+\delta X\}$ soluzione (approssimata) dell'equazione $\max_{\{\delta x\}} E(\{W\},\{X+\delta X\})$ (il calcolo contestuale dei migliori valori dei parametri per minimizzare l'errore e delle variazioni dei dati di addestramento per massimizzare l'errore è chiamato un "*problema min-max*"). Il risultato è espresso nella seguente tabella (i numeri indicano la percentuale di errore espressa con un numero tra 0 e 1 per i tre set di dati *Train*, *Test*, *Adv*, per ogni ciclo/epoca di addestramento) [Rif. 8].

Epoca	Train Err.	Test Err.	Adv Err.
0	0.272300	0.031000	0.666900
1	0.026417	0.022000	0.687600
2	0.017250	0.020300	0.601500
3	0.012533	0.016100	0.673000
4	0.009733	0.014400	0.696600
5	0.003850	0.011000	0.705400
6	0.002833	0.010800	0.696800
7	0.002350	0.010600	0.707500
8	0.002033	0.010900	0.714600
9	0.001783	0.010600	0.708300

Tavola 1 [fonte Rif. 8]

Al termine dell'addestramento, l'errore sul set di dati *Train* è inferiore allo 0,2%, sul set di dati *Test* (su cui non viene fatto addestramento) è circa dell'1% e il programma ha costruito un set di dati di *Adversarial Examples* (*Adv*) molto simile al set di dati *Train,* ma su cui il modello ML sbaglia il riconoscimento dell'immagine il 71% delle volte.

Cosa è cambiato nelle immagini *Adversarial Examples* rispetto alle immagini del set di *Train*? Spesso la modifica corrisponde all'aggiunta di "rumore bianco" all'immagine che può rendere un po' sfocati alcuni particolari e introdurre dei piccoli punti con colori errati in maniera apparentemente casuale (difetti dell'immagine o "*glitch*"), simile a quanto visto in Fig. 2. Nulla, comunque, che renda all'occhio umano la nuova immagine realmente diversa da quella originale.

Ora è possibile addestrare il modello ML anche utilizzando il set di dati di *Adversarial Examples* (*Adv*) che è stato generato, ma utilizzando sempre il set di dati *Test* solo per verificare l'addestramento. Il risultato è:

Epoca	Train Err.	Test Err.	Adv Err.
0	0.715433	0.068900	0.170200
1	0.100933	0.020500	0.062300
2	0.053983	0.016100	0.046200
3	0.040100	0.011700	0.036400
4	0.031683	0.010700	0.034100
5	0.021767	0.008800	0.029000
6	0.020300	0.008700	0.027600
7	0.019050	0.008700	0.027900
8	0.019150	0.008600	0.028200
9	0.018250	0.008500	0.028300

Tavola 2 [fonte Rif. 8]

Gli errori finali sui set di dati di *Train* e di *Test* sono cambiati ma le differenze non sono significative per l'esercizio svolto, mentre l'errore sul nuovo set di dati di *Adversarial Examples* (*Adv*) generato è sceso dal 71% a circa il 2,8%. Il modello ML ha quindi imparato a riconoscere correttamente anche gli *Adversarial Examples* generati con questo procedimento e che ora non sono più "antagonisti". Un modello ML che in pratica non ha *Adversarial Examples* è chiamato "**robusto**"; quindi questo modello ML può dirsi "robusto" almeno sino a quando non sarà trovata una nuova classe di suoi *Adversarial Examples*.

È importante sottolineare che dopo aver ri-addestrato il modello utilizzando anche il set di *Adversarial Examples*, si può generare con lo stesso procedimento un nuovo set di dati massimizzando l'errore,

ma l'errore rimane molto piccolo e il modello riconosce correttamente anche questo set di dati. In altre parole, il procedimento adottato in questo esempio particolare per costruire un set di *Adversarial Examples* funziona solo la prima volta, poi non genera più set di *Adversarial Examples*.

Riassumendo, è possibile minimizzare gli errori sui dati di addestramento calcolando il valore dei parametri {W} e al contempo generare un set di dati {X+δX} quasi indistinguibile dal set di dati di addestramento ma che massimizza l'errore del modello ML. Anche se poi è possibile ri-eseguire il processo di addestramento includendo gli *Adversarial Examples* individuati creando un modello ML robusto, ad oggi non è noto un metodo per generare **tutti** gli *Adversarial Examples* di un modello ML, ovvero garantire che gli *Adversarial Examples* trovati siano **tutti** quelli di un modello ML. Inoltre, come indicato precedentemente, aggiornare l'addestramento di un modello ML può portare a nuovi problemi, ad esempio la "smemoratezza" (*Forgetfulness,* dimenticare informazioni precedentemente acquisite) o una maggiore predisposizione ad attacchi di inversione e estrazione di dati di addestramento, problematiche rilevate in alcuni modelli ML di studio.

Il messaggio che viene da questo esercizio è che il processo di addestramento attuale non è completo: l'esistenza degli *Adversarial Examples* non è dovuta a qualche disattenzione o peculiarità di alcuni modelli ML particolari, ma è una caratteristica di come sono costruiti i modelli ML attuali. La ricerca scientifica su questo problema è molto intensa [Rif. 1], sono stati introdotti anche standard e *benchmark* sulla "robustezza" dei modelli ML e sui metodi per renderli robusti [Rif. 9], ma è probabile che per poter trovare una soluzione definitiva sia necessaria una migliore comprensione (soprattutto matematica) dei modelli di *Machine Learning*.

Riferimenti Bibliografici

Rif. 1: N. Carlini, "*A Complete List of All (arXiv) Adversarial Example Papers*", https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html , nel solo anno 2022 risultano pubblicati quasi 3.000 articoli scientifici relativi a "*Adversarial Machine Learning*"

Rif. 2: T. Gu, B. Dolan-Gavitt, S. Garg, "*BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain*", 2017, arXiv:1708.06733

Rif. 3: I.J. Goodfellow, J. Shlens, C. Szegedy, "*Explaining and Harnessing Adversarial Examples*", 2014, arXiv:1412.6572

Rif. 4: B. Biggio, B. Nelson, P. Laskov, "*Poisoning Attacks against Support Vector Machines"*, 2013-03-25, ArXiv:1206.6389;

B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, F. Roli, Fabio, "*Evasion attacks against machine learning at test time"*, 2013, ECML PKDD. Lecture Notes in Computer Science. Vol. 7908. Springer. pp. 387–402, arXiv:1708.0613;

B. Biggio, F. Roli, Fabio, "*Wild patterns: Ten years after the rise of adversarial machine learning"*, dicembre 2018, Pattern Recognition. 84: 317–331, arXiv:1712.03141

Rif. 5: M. Sharif, S. Bhagavatula, L. Bauer, M.K. Reiter, "*Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition*", https://users.cs.northwestern.edu/~srutib/papers/face-rec-ccs16.pdf

Rif. 6: Microsoft "*Threat taxonomy - Failure modes in machine learning*", https://learn.microsoft.com/en-us/security/engineering/

Rif. 7: MITRE *"Adversarial Threat Landscape for Artificial-Intelligence Systems (ATLAS)"*, https://atlas.mitre.org/

Rif. 8: Z. Kolter, A. Madry, "*Adversarial Robustness - Theory and Practice*", https://adversarial-ml-tutorial.org/

Rif. 9: F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, M. Hein, "*RobustBench: a standardized adversarial robustness benchmark*", https://robustbench.github.io/ , arXiv:2010.09670

ICT Security MAGAZINE

ISCRIVITI ALLA NEWSLETTER

per ricevere aggiornamenti sulle prossime iniziative. Seguici sui canali social: **Linkedin**, **Facebook**, **Twitter**